EP 33612 ②

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 September 2001 (07.09.2001)

PCT

(10) International Publication Number
WO 01/65417 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US01/06534

(22) International Filing Date: 28 February 2001 (28.02.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/186,554	2 March 2000 (02.03.2000)	US
09/709,070	8 November 2000 (08.11.2000)	US

(71) Applicant: IDINI CORPORATION [US/US]; 1346 Ridder Park Drive, San Jose, CA 95131 (US).

(72) Inventor: LIN, Chris, H., D.; 1888 Traden Drive, San Jose, CA 95132 (US).

(74) Agents: SMITH, Albert, C. et al.; Fenwick & West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).

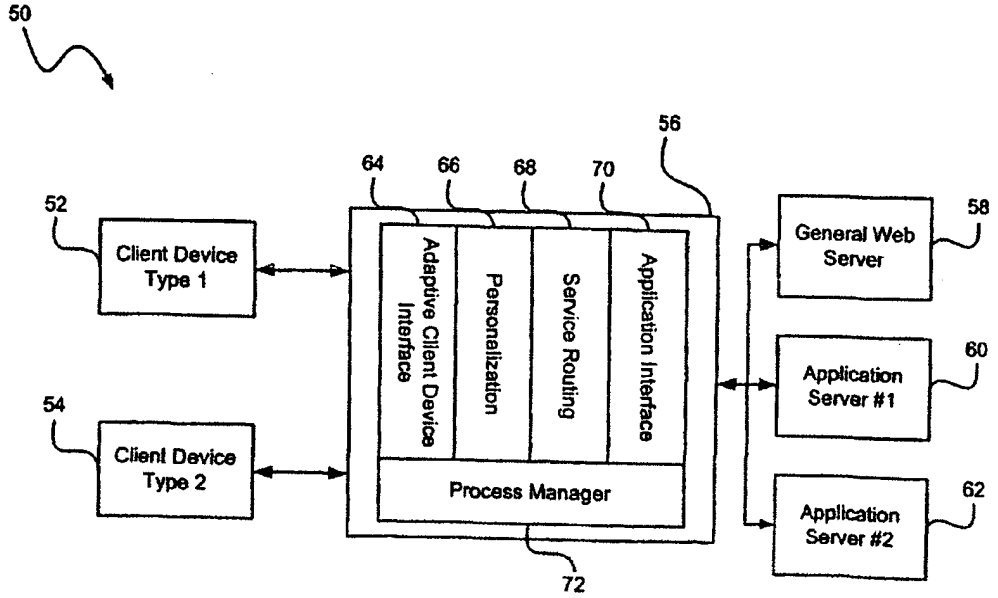
(81) Designated State *(national)*: JP.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: IMPROVED DEVICE INDEPENDENT REMOTE DATA MANAGEMENT



(57) Abstract: A method and apparatus for remotely controlling server command execution by a client computer are presented. The client is connected through a first server (58) to a plurality of servers. The client communicates with the first servers (58) within the plurality of servers using the language necessary to accomplish the desired task. The language used is determined using the operating environment and the application program interface (70) receiving the command.

WO 01/65417 A1

SPECIFICATION
IMPROVED DEVICE INDEPENDENT REMOTE DATA MANAGEMENT

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to electronic devices used to interact with information stored on a server. More particularly, the present invention provides an apparatus interface between client devices and network servers which allows applications executed on network server to be remotely controlled and manipulated by the client device.

2. The Background Art

In modern computing, it is commonplace for electronic devices to be linked together in a network and thus be able to share and operate on information contained within that network. Networks include server devices which have information and application software which may be used to "serve" the needs of client devices.

It is known that a given network device may at times act as either a client device or a server device. In this specification, the conventional definitions of server and client devices are retained, and those terms shall refer to devices which,

although possibly capable of alternating between server and client modes, are operating in the named mode at the time a described activity is taking place.

Typical networks contain a wide variety of client device types, ranging from larger devices such as personal computers and terminals, to smaller devices such as Personal Digital Assistants (PDA's), and network capable cellular telephones.

Each different client device type typically has a different language associated with it, due to the variety of different functions those client device types may be designed to perform.

In order to serve a given client device type, a server is required to store data which would be used by that client device type in the language understood and used by that device type. Examples of those languages include WML, HTML, etc. Further, the server is required to understand the syntax of the commands sent to it by client devices, and respond in kind so that the client device will understand the response.

In addition to certain client device types being incapable of processing information which is in a different language than that used by the client device, the client application software which is available to interact with the server

information often has functionality which is substantially reduced from that functionality typically seen in modern desktop computers, due to the client device having much less display area, lower power consumption requirements, less processing power, and less memory. Thus, it is typical that a substantially reduced subset of commands is available to those client device types.

Examples of prior art networks are seen in FIGS. 1A, 1B, and 1C.

Referring to FIG. 1A, a first client device 10 is linked to server 12 which contains information files 14 and 16 in wireless markup language (WML), the language used by client device 10.

Referring to FIG. 1B, a second client device 18 is linked to server 20 which contains files 22 and 24 in hypertext markup language (HTML) format, the language used by client device 22.

Referring to FIG. 1C, a network 30 may contain a first client device 32, and a second client device 34, both linked to server 36. Server 36 contains files 38, 40, and 42 in WML, HTML, Xtensive markup language (XML) respectively, the particular languages suited for particular client devices which are expected to occasionally require the information stored therein. File 44 is in text format, and

may be a file which is incorporated at runtime into a display controlled by files 38, 40, or 42.

In this FIG. 1C example, files 38, 40, 42 and 44 may contain the same information, but in different formats. Thus, if client device 32 requires files in WML format and requests information contained in the group of files represented by files 38, 40, 42 and 44, server 36 must select file 38 for distribution to device 32. Correspondingly, if device 34 is requires files in XML format and requests information contained in the group of files represented by files 38, 40, 42 and 44, server 36 must select file 42 for distribution to device 34.

Prior art networks, while suited for their intended purpose of delivering information to client devices in the formats well-suited for those devices, suffer significant drawbacks in that it is necessary that servers such as server 36 contain desired information in the many formats desired by the different client devices which it serves. Thus, an individual document is required to be stored in as many formats as the number of types of client devices allowed to be connected to the network.

It would therefore be beneficial to provide a method and apparatus which requires only one copy of a given piece of information in a format known to the server but which can also provide that information in any format required by the

various types of portable electronic devices which may become linked to the network.

Further, prior art networks suffer from their ability to allow client devices to remotely control the manipulation of files. It was previously stated that client devices have limited processing power and are therefore limited in the application software that they can execute. It would therefore be tremendously beneficial to provide a system and method for client devices to be able to manipulate information in many different formats and using multiple languages.

SUMMARY OF THE INVENTION

A method and apparatus for remotely controlling server command execution by a client computer are presented. The client is connected through a first server to a plurality of servers. The client communicates with the first server using a first language, and the first server communicates with each of the servers within the plurality of servers using the language necessary to accomplish the desired task. The language used is determined using the operating environment and the application program interface receiving the command. In one embodiment, the first server consults a rule identify a set of server commands in a second language which is equivalent to the client command, and then consults a capability table to determine one or more servers configured to execute each command. The respective servers are then caused to execute one or more of the commands with the set of server commands. Any results from those command executions are then translated back into the client language and transmitted to the client for processing.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. 1A, 1B, and 1C are examples of prior art networks.

FIG. 2 is one example of a electronic system according to the present invention wherein client devices are linked to servers.

FIGS. 3A and 3B are a flow chart showing a method for processing client device commands in a network according to the present invention.

FIG. 4 is an example of a rule table according to the present invention.

FIG. 5 is a flow chart showing another method according to the present invention.

FIG. 6 is an example of a capability table according to the present invention

FIG. 7 is an example of a present invention server loading table.

FIGS. 8A and 8B are a flow chart showing another method of the present invention.

FIGS. 9A, 9B and 9C are a flow chart showing yet another method of the present invention.

FIGS.10A, 10B and 10C are a flow chart showing yet another method of the present invention.

DETAILED DESCRIPTION OF ONE EMBODIMENT

Those of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to such skilled persons having the benefit of this disclosure.

The present invention provides a method and apparatus for allowing client devices having limited command functionality and which utilize a first language to control the execution of application programs on remote servers using a second language.

Briefly, a client device is connected to a second server through a present invention first server. Commands are issued by the client to the first server in a

first command language such as WML. If the command relates to an application provided by a second server, the first server translates the command into the language and format used by the second server, and transmits that command to the server for action. Following the execution of the command by the second server, the results are transmitted by the second server to the first server. Depending on what those results are, the first server may transmit the results to the client device.

In this manner, a limited functionality client device may remotely cause applications to be executed on a server which operates using a command syntax or language unfamiliar to the client device.

FIG. 2 is one example of a present invention electronic system wherein client devices are linked to servers.

Referring to FIG. 2 network 50 includes client devices 52 and 54 which are of different types and therefore have different command syntax and use different languages, adaptive server 56, and servers 58, 60, and 62. Adaptive server 56 includes an adaptive client interface 64, a personalization module 66, a service routing module 68, and application 70. Client devices 52 and 54 and servers 58, 60, and 62 are typical prior art devices.

Server 58 is an example of a typical prior art web server. Servers 60 and 62 are examples of typical prior art application servers. For the purposes of this disclosure, servers 50, 60, and 62 use different command syntax and languages.

Client interface 64 receives input information from either of clients 52 or 54. Interface 64 is adaptive in that it is configured to determine the type of devices connected at different ports, and communicate with those devices in their own desired command syntax. Therefore, by way of example, a first device may use WML, and a second device may use HTML, but commands issued by either device through interface 64 will be understood by interface 64. Each port on the interface may be attached to any type of device.

Personalization module 66 is configured to personalize each networking session with each client. Following an authentication session which may be managed by either interface 64 or personalization module 66, a preferences set is consulted to determine the various settings the particular user of the client device prefers. Methods to personalize network sessions are known in the art.

Service routing module 68 determines the proper resources with which to handle command requests submitted by client devices. After a command is initiated on a client device and transmitted to server 56, routing module 68 determines the proper server resources to utilize to accomplish the desired tasks,

and then routes the requests to those servers. In order to determine the resources, routing module 68 examines a rule table to identify the proper server commands to satisfy particular client command requests.

Once the proper server commands are identified, routing module 68 examines a capability table and a loading table to determine which servers are configured to perform the needed commands, and ranks those properly configured servers in order of their loading. Generally, the most available server having the quickest anticipated execution time will be chosen to execute the commands, and the commands will be particularly structured and formatted for that server.

Application interface 70 is configured to transmit the proper server commands and handle all of the communications between server 56 and servers 58, 60, and 62. Those of ordinary skill in the art will readily appreciate that each application program has a set of commands associated with it. Those commands are typically received through the use of an application program interface (API) which has a format associate therewith. It is this format, coupled with any special method of transferring data to the server, that is termed "language" in this specification. The term "language" as used herein is therefore meant to be broad, rather than limiting.

Process manger 72 is configured to manage events and processes within server 56.

FIGS. 3A and 3B comprise a flowchart showing a method of the present invention.

In describing methods of the present invention herein, reference is made to a first server and a second server. It is intended that the term "first server" apply to a server such as server 56 (FIG. 1) through which client commands are passed in order to be processed and translated from the client device language into a server language understood by a second server. It is therefore intended that the term "second server" apply to those servers such as servers 58, 60, and 62 which receive translated commands from the first server.

Referring to FIGS. 3A and 3B together, the method begins at block 80 where a client device initiates a command. Such a command is not restricted and can therefore be an application command, a web display command, or any other commands used in a network.

At block 82, the client device transmits that client command to a present invention first server.

At block 84, the first server receives the client command and consults a rule table to determine a set of one or more server commands equivalent to the client command. An example of such a rule table is seen in FIG. 4 where a first column of commands 86 translates into a set of one or more commands in a second column of commands 88. By way of example and not intended to be limiting, a client "compare" command 90 translates into a set of server commands including commands 92, 94, 96, 98, 100, 102, and 104.

The server commands equivalent to "compare" command 90, in this example, would be a set of commands which include the steps of (92) determining whether the files being compared are in the same file format, (94) if the files being compared are not the same file format, converting one document into the other documents format, or converting both documents into a common third format, (96) opening the first file, (98) opening the second file, (100) comparing the two files, (102) closing the first file, and then (104) closing the second file. Those of ordinary skill in the art will readily recognize that commands such as those described herein may be presented in one or more alternate sequences and still accomplish an equivalent result.

The foregoing example is meant to be illustrative only. Those of ordinary skill in the art will readily recognize that many different command constructs will

achieve the same result. It is intended that all such constructs be included in the present invention.

Returning to the discussion of FIG. 3, at block 110, the first server chooses, in order of required execution, a command from the set of server commands previously determined at block 84.

At block 112, the first server translates the chosen command into the language (or format) required by the second server and transmits that translated command to the second server for execution.

At block 114, the second server executes the translated command and returns any results back to the first server.

At block 116, it is determined whether all commands from the set of server commands have been processed through blocks 110, 112, and 114. If yes, the method proceeds with block 118 where the first server collates the results of the various command executions and translates those results into the first language (or format) and transmits the translated results to the client for display.

If, at block 116, not all commands from the set of server commands had been processed, the method proceeds at block 110 where a new command is chosen from the set and processed.

In the preceding example, a a client command is translated into a set of server commands, and those server commands are then processed in the order with which those commands are expected to be executed. As each command is processed, it is transmitted to the proper server for execution. Those of ordinary skill in the art will readily recognize that the commands making up the set of server commands identified as being equivalent to the client command may be occasionally be processed in parallel, and alternatively in different order, and still achieve an equivalent result. Further, rather than transmitting each command as it is processed, all commands may be processed and then transmitted to the respective servers as a set.

A more detailed presentation of one method contemplated by the inventor to accomplish the execution of block 84 is seen in FIG. 5.

Prior to the execution of block 84, the first server has received a client command from the client device.

Referring to FIG. 5, the detailed method of block 84 begins at block 130 where the first server interrogates a rule base such as seen in FIG. 4 in order to map the client command into a set of equivalent server commands.

At block 132, the first server routing module chooses one of the server commands from the set of equivalent server commands to be processed. It is contemplated that each of the server commands will be chosen in the desired order of execution, but it is not necessary that the order of command processing be chosen in that manner.

Those of ordinary skill in the art having the benefit of this disclosure will readily recognize that this example method operates on the commands in order, and transmits translated commands to the second server as they are translated. However, an alternate embodiment may cause the commands to be translated in any order, cache the results, then transmit the translated commands in their correct order.

At block 134, the first server routing module interrogates a capabilities list to determine one or more servers which are configured to execute the required server commands. One example of such a capabilities table is seen in FIG. 6 which includes a column 136 of commands, a column 138 of applications, and a column 140 of servers being configured to execute those applications..

By way of example, assume that the client command is to convert a file from a first file format to a second file format. The client command is transmitted from the client device in the client language to the first server which translates that client command into one or more server commands using the rule base. The capabilities table shows two servers having IDs 129 and 135 configured to execute the "convert" command.

After determining that there are one or more servers configured to execute the command, a server loading table such as that depicted in FIG. 7 is interrogated to determine which of those two servers can execute the command most efficiently. In this example, and using the capabilities table of FIG. 7, since the loading on server 129 is less than the loading on server 135, server 129 will be chosen to execute the command.

At block 150, it is determined whether each server command has been mapped to a server configured to execute that command.

If yes, the method proceeds with block 152 where it is determined whether all server commands have been processed through blocks 132, 134, and 150. If yes, the method ends.

If, at block 150, there exists at least one server command which doesn't have an associated server configured to execute that command, the routing module, at block 154, identifies a server capable of being configured to execute the command and configures that server. This process of identifying a potential server and properly configuring it is further described in U.S. Patent Application S/N 09/432,491 entitled :"DIRECTORY-BASED FAILURE RECOVERY AND LOAD BALANCING SYSTEM" naming inventor Horng-Dar Lin and assigned to iDini, Inc. which is hereby incorporated herein in its entirety.

A further illustrative method of the present invention is presented herein as FIGS. 8A and 8B.

Referring to FIGS. 8A and 8B together, the method begins at block 160 where a client device such as device 52 (FIG. 2) initiates a document display request and transmits that request to a first server of the present invention such as server 56. By way of example, assume that the client device command is to present the details of a word-processing document to the client device for viewing. Since the client device doesn't have the processing power to run such a complex application such as WordPerfect, it is necessary that the first server ensure that the WordPerfect application be executed on a server (its own, or a second server) with the proper document requested by the client device and that display details are translated and forwarded to the client device.

At block 162 the first server receives the block 160 request. It is assumed herein that information describing characteristics of device 52 have previously been received, and that the client device type is known by the present invention first server. It is therefore not necessary that information describing characteristics the devices such as device 52 be transmitted with every document display request.

However, in order for a server such as server 56 to be able to provide display, edit or other device dependent services, it is necessary that such information describing characteristics of device 52 be provided at some point prior to server 56 providing display information back to device 52. In the absence of specific client device characteristic information, server 56 may assume a particular default set of characteristics that would be used absent information pertaining to the specific device such as device 52.

At block 164, the present invention first server identifies the language required to retrieve the document and requests the document from its storage location using that identified language. The identification of the language required is accomplished by determining the type of server on which the document resides, together with identifying the document file format. Those of ordinary skill in the art are readily aware of many ways to identify a server type and a file format type.

At block 166. the first server determines the content of the information stored in the retrieved document and isolates differing content types into content buckets. Examples of content types include images, text, graphics, etc.

At block 168, the first server transmits overview details of each content bucket to the requesting client device, based on preference settings. If, for example, a preference setting shows that the client device associated with a given login identifier prefers text over graphics, or summary information instead of actual text, the information provided the client at this block is adjusted accordingly.

At block 170, the client device displays details relating to the various content buckets, and a particular content bucket is chosen for display by transmitting a "display bucket" command to the first server.

At block 172, the present invention first server compares information about the particular content bucket with the display characteristics of the client device to determine whether all of the information from the desired content bucket can be displayed on a single screen of that client device.

If no, at block 174 a subset of information from the content bucket is selected for display by the first server. This subset may include any set of

information within the content bucket which may be displayed in a single screen of the client device, such as the upper right region of the content bucket, the upper left region of the content bucket, etc.

The selected subset of information selected for display is then transferred to the client device and displayed. Once the information has been transferred, the first server process manager awaits the receipt of the next client device command. Those of ordinary skill in the art having the benefit of this disclosure will readily recognize that the client device may transmit windowing commands which cause the first server to refresh the client device display to include different portions of the chosen content bucket.

At block 176, a nonwindowing command is transmitted by the client device to the first server and a determination is made whether the command is to choose a new content bucket for display. If yes, the method proceeds at block 172. If no, the method proceeds at block 178 where the new command is evaluated and processed as described herein.

Those of ordinary skill in the art having the benefit if this disclosure will readily recognize that a method for using the client device to view a document which is stored on a remote server may be as described above, where the document is retrieved by the first server and then presented to the client device one

display at a time, or may instead be a method wherein the document is left on the remote server and that remote server is remotely controlled by the first server as described herein to provide data relating to the different content buckets.

FIGS. 9A, 9B, and 9C together are a flow chart showing a method of the present invention.

Referring to FIGS. 9A, 9B, and 9C together, the method begins at block 200 when a client communication is received by a first server implementing the present invention such as server 56 (FIG. 2).

At block 202, any necessary parsing of the client communication is performed. If the client communication is a single command, no parsing is required, and therefore the block 202 operation isn't performed. However, if the client communication received at block 200 contains a plurality of commands, the client communication is parsed into a set of distinct first server commands.

At block 204, a command to be processed is chosen from the set of first server commands created at block 202. It is contemplated that some first server commands will be dependent upon the prior execution of other commands, while other first server commands are not dependent upon other commands and may therefore be executed at any time. It is contemplated that this block 204 operation

will take into account the commands available for processing, and the commands previously executed, as is necessary.

Once a first server command is chosen for processing, a rule table such as that seen in FIG. 4 is searched, at block 206, for one or more application procedures that are equivalent to that chosen first server command. Recall that a client communication will typically include information relating to the operation desired, the format of any documents involved, and the type of client requesting the service. The rule table, as seen in FIG. 4, includes a command and its argument(s), and procedures and arguments which an application server may perform that are equivalent to the desired first server command. Optionally, the rule table includes a priority indicator which indicates the order in which similar procedures shall be processed. In the example of FIG. 4, no specific priority indicator is shown. However, the rules are processed in order of occurrence in table.

For example, if the first server command is an edit command, a rule table inquiry provides procedures 208 and 210, procedure 208 having the steps of open (212), edit command (214), and close (216). Therefore, to edit a document having format X, it is required (under this procedure) to open the document, perform the desired edit command, and then close the document.

If the first procedure isn't able to be performed, the second procedure may be performed, the second procedure having the steps of converting the document from format X to format Y (218), performing the desired edit command in format Y (220), then converting the resulting document back into format X (222). Those of ordinary skill in the art having knowledge of this disclosure will readily recognize that this second procedure requires multiple rule table interrogations, and recursion.

Recursion is used when a command such as command 220 is also a rule, such as rule 208. In this instance, when command 220 is processed, there may be no application servers that understand how to perform the command directly. When this situation occurs, the rule table is examined again to determine if a equivalent procedure is available. At that time, procedure 208 will be substituted for command 220.

If the rule table optionally includes a priority indicator for one or more procedures, that priority indicator will be utilized when choosing which procedure to process. For example, although procedures 208 and 210 are both "edit" procedures, a system may exist wherein it is highly desirable to perform procedure 210 prior to executing procedure 208, then only executing procedure 208 as necessary. Therefore, the priority indicator will indicate that procedure 210 is a higher processing priority than procedure 208.

At block 224, one of the procedures determined at block 206 is chosen for processing. It is contemplated that the procedures in the rule table will be listed in order of their desirability for execution. Therefore, procedure 208 would be attempted prior to procedure 210, even though both procedures are equivalent to the same rule. Those of ordinary skill in the art familiar with this disclosure would be capable of determining other methods to cause highly preferred procedures to execute prior to equivalent less-preferred procedures.

At block 226, the procedure determined at block 224 is examined, and a command within that procedure is chosen for processing. Although it is contemplated that commands within procedures will often be executed in the order that they are listed in the rule table, it is contemplated that some commands having no dependencies will execute concurrently or out of order with other commands listed in the same procedure.

At block 228, the various application servers are polled to determine which application servers are configured to execute the procedure command chosen at block 226.

At block 230, it is determined whether at least one application server is configured to perform the chosen procedure command. If yes, the method

proceeds to block 232 where it is determined whether the chosen procedure command depends on the results of a previous command in order to be executed properly.

If no, the method proceeds at block 234 where the most efficient application server is chosen to execute the chosen command, and that server is then caused to execute that command. At block 236, it is determined whether the chosen procedure comprises commands which have not yet been executed. If not, the method proceeds at block 238 where it is determined whether any unprocessed first server commands exist.

If no unprocessed first server commands exist, the method ends.

If, at block 230, there wasn't at least one application server configured to execute the desired application command, the method proceeds at block 240 where at least one application server is configured to execute that command. Further details on at least one method which may be used to configure such an application server is described in U.S. Patent Application S/N 09/432,491 entitled :"DIRECTORY-BASED FAILURE RECOVERY AND LOAD BALANCING SYSTEM" naming inventor Horng-Dar Lin and assigned to iDini, Inc. After a server is configured to execute the desired command, the method proceeds at block 232.

If, at block 232, it is determined that the chosen application server command requires the results of another command execution in order to itself execute properly, it is determined, at block 242, whether that other command has completed. If not, the method proceeds at block 244 where execution of the present command is delayed until the precedent command has completed execution. The method then proceeds at block 234.

If, at block 236, unexecuted commands within the chosen procedure exist, one of those executed commands is chosen for execution as previously described, and the method proceeds with block 228.

If, at block 238, unprocessed procedures exist, one of those procedures is chosen for processing as previously described, and the method proceeds with block 206.

FIGS. 10A, 10B and 10C are a flow chart showing yet another method of the present invention.

In the method of FIG. 9, a client command was received, and that client command was compared against a rule table to determine procedures that are

equivalent to the client command, and those procedures were executed . One of those procedures was chosen for processing based upon a predetermined priority, and commands within that procedure were then executed on application servers configured for that purpose..

In the FIG. 10 embodiment, a similar order of operations takes place. However, following the step where all procedures equivalent to a given client command are determined, the application servers are polled to determine two things. First, it is determined whether one or more application servers are configured to execute the desired commands. Second, data corresponding to those configured servers if obtained in order for a decision to be made as to the most efficient server to use to execute the given commands within a given procedure. Once it is determined which procedure is most efficient to execute on particular configured servers, the various commands within those procedures are executed.

Referring to FIGS. 10A, 10B, and 10C together, the method begins at step 250 when a client communication is received by a first server implementing the present invention such as server 56 (FIG. 2).

At block 252, any necessary parsing of the client communication is performed. If the client communication is a single command, no parsing is required, and therefore the block 252 operation isn't performed. However, if the

client communication received at block 250 contains a plurality of commands, the client communication is parsed into a set of distinct first server commands.

At block 254, a command to be processed is chosen from the set of first server commands created at block 252. It is contemplated that some first server commands will be dependent upon the prior execution of other commands, while other first server commands are not dependent upon other commands and may therefore be executed at any time. It is contemplated that this block 254 operation will take into account the commands available for processing, and the commands previously executed, as is necessary.

Once a first server command is chosen for processing, a rule table such as that seen in FIG. 4 is searched, at block 256, for one or more application procedures that are equivalent to that chosen first server command. The same or similar criteria is used here as was previously described for block 206 (FIG. 9).

At block 258, the various application servers are polled to determine their configuration and available for processing commands within the procedures from block 256. It is contemplated that the various application servers will respond with their configuration information, including application loading. Therefore, using the loading information, an intelligent decision may be made about which

configured server to use to execute a given command, and which server(s) to configure to execute commands it is not yet configured to execute.

At block 260, it is determined whether at least one server is configured to execute a procedure commands which are (as a set) equivalent to the client command. It is possible that one server will be able to perform all of the commands within a given procedure. However, it is contemplated that two or more servers will be used, each server performing a subset of the commands within a given procedure.

If at least one server is configured to perform at least one equivalent procedure, at block 262, it is determined whether multiple equivalent procedures able to be performed which are equivalent to the client command being processed. If yes, the method proceeds at block 264 where one of those procedures are chosen for execution. It is contemplated that server efficiency will be taken into account when choosing the procedure to be utilized, and the servers on which commands within that procedure will be executed. Those of ordinary skill in the art readily recognize that server efficiency includes many factors, including loading, configuration time, etc.

At block 266, the commands within the chosen procedure are issued to the various servers for execution, taking into account command precedence as previously discussed in relation to FIG. 9.

If, at block 260, none of the equivalent procedures may be executed because either no servers are configured to execute one or more commands within the procedures or the servers that are configured properly are overloaded, the method proceeds at block 268 where one or more procedures are chosen to be configured. It is contemplated that a decision to configure one procedure may be made, or a decision to configure two or more equivalent procedures may be made, based on command execution history, priority, and other factors known to those of ordinary skill in the art.

At block 270, each command with the chosen one or more procedures that isn't currently configured on an applications server is configured. It is contemplated that some commands that are currently configured to be executed on a server may be configured to also be executed on a different server, in order to improve execution efficiency.

At block 266, the commands within one of the equivalent procedures are executed. It is contemplated that if multiple servers had been configured at block

270 so that more choices were available, the method may proceed again at either block 258 or block 262, depending on a given system designers desires.

If, at block 262, only one equivalent procedure was properly configured for execution, that procedure is designated, at block 272, as the chosen procedure. The method then proceeds at block 266, where commands within that procedure are executed.

While embodiments and applications of this invention have been shown and described, it would be apparent to those skilled in the art that many more modifications than mentioned above are possible without departing from the inventive concepts herein. The invention, therefore, is not to be restricted except in the spirit of the appended claims.

What is Claimed is:

1. In a networked computer system having a client connected through a first server to an application server, the client being configured to communicate with said first server using a first language, the first server and the application server configured to communicate with each other using a second language, a method for remotely handling application commands comprising:

initiating at least one command in said first language on the client and transmitting said at least one command to said first server as a command string;

parsing said command string into distinct first server commands;

choosing from said first server commands a first command to be processed;

translating, using a rule table, said first command in said first language into a procedure comprising at least one equivalent command in said second language;

choosing from said equivalent commands, a desired command to be executed;

choosing an application server configured to execute said desired command;

causing said application server to execute said desired command.

2. The method of claim 1 further including

determining that an application server is not configured to execute the desired command;

configuring an application server to properly execute the desired command.

3. The method of claim 2 wherein determining that an application server is not configured to execute the desired command is accomplished by polling the available application servers to determine whether the desired command is available for execution, and receiving information that no server is configured to execute the desired command.

4. The method of claim 1 wherein said rule table is stored entirely on said first server.

5. In a networked computer system having a client connected through a first server to an application server, the client being configured to communicate with said first server using a first language, the first server and the application server configured to communicate with each other using a second language, a method for remotely handling application commands comprising:

initiating at least one command in said first language on the client and transmitting said at least one command to said first server as a command string;

parsing said command string into distinct first server commands;

choosing from said first server commands a first command to be processed;

translating, using a rule table, said first command in said first language into a procedure comprising at least one equivalent command in said second language;

choosing from said equivalent commands, a desired command to be executed;

determining that the application servers configured to execute the desired command cannot perform the command within a predetermined acceptable time span;

configuring an additional application server to properly execute the desired command.

causing said additional application server to execute said desired command.

6. In a networked computer system having a client connected through a first server to at least three application servers, the client being configured to communicate with said first server using a first language, the first server and said at least three application servers configured to communicate using a second language, a method for remotely handling application commands comprising:

initiating at least two commands in said first language on the client and transmitting said at least two commands to said first server;

choosing from said at least two first server commands at least two commands to be processed;

translating said at least two command in said first language into an equal number of equivalent command sets in said second language;

choosing from said equivalent command sets, a first command set to be executed;

causing a first one of said application servers to execute said first command set;

choosing from said equivalent command sets, a second command set to be executed;

causing a second one of said application servers to execute said second command set.

7. The method of claim 6 wherein said causing a first one of said application servers step is further defined as

determining the server efficiency factors for at least two of said at least three application servers;

choosing the application server having the most desirable server efficiency factor to execute said first command set;

causing said chosen server to execute said first command set.

8. The method of claim 6 wherein said causing a second one of said application servers step is further defined as

determining that the server efficiency factors for a second application server is more desirable than the server efficiency factor for said first application server;

causing said second application server to execute said second command set.

9. The method of claim 6 wherein said translating is accomplished using a rule table.

10. A machine readable media having stored thereon machine executable instructions to perform a method in a networked computer system having a client connected through a first server to an application server, the client being configured to communicate with said first server using a first language, the first server and the application server configured to communicate with each other using a second language, the method being for remotely handling application commands comprising:

initiating at least one command in said first language on the client and transmitting said at least one command to said first server as a command string;

parsing said command string into distinct first server commands;

choosing from said first server commands a first command to be processed;

translating, using a rule table, said first command in said first language into a procedure comprising at least one equivalent command in said second language;

choosing from said equivalent commands, a desired command to be executed;

choosing an application server configured to execute said desired command;

causing said application server to execute said desired command.

11. The machine readable media of claim 10 further having stored thereon instructions to perform the following additional instructions:

determining that an application server is not configured to execute the desired command;

configuring an application server to properly execute the desired command.

12. The machine readable media of claim 11 wherein determining that an application server is not configured to execute the desired command is accomplished by polling the available application servers to determine whether the desired command is available for execution, and receiving information that no server is configured to execute the desired command.

13. A machine readable media having stored thereon machine executable instructions to perform a method in a networked computer system having a client connected through a first server to an application server, the client being configured to communicate with said first server using a first language, the first server and the application server configured to communicate with each other using a second language, the method being for remotely handling application commands comprising:

initiating at least one command in said first language on the client and transmitting said at least one command to said first server as a command string;

parsing said command string into distinct first server commands;

choosing from said first server commands a first command to be processed;

translating, using a rule table, said first command in said first language into a procedure comprising at least one equivalent command in said second language;

choosing from said equivalent commands, a desired command to be executed;

determining that the application servers configured to execute the desired command cannot perform the command within a predetermined acceptable time span;

configuring an additional application server to properly execute the desired command.

causing said additional application server to execute said desired command.

14. A machine readable media having stored thereon machine executable instructions to perform a method in a networked computer system having a client connected through a first server to at least three application servers, the client being configured to communicate with said first server using a first language, the first server and said at least three application servers configured to communicate using a second language, the method for remotely handling application commands comprising:

initiating at least two commands in said first language on the client and transmitting said at least two commands to said first server;

choosing from said at least two first server commands at least two commands to be processed;

translating said at least two command in said first language into an equal number of equivalent command sets in said second language;

choosing from said equivalent command sets, a first command set to be executed;

causing a first one of said application servers to execute said first command set;

choosing from said equivalent command sets, a second command set to be executed;

causing a second one of said application servers to execute said second command set.

15. The machine readable media of claim 14 further having stored thereon instructions to perform the following additional instructions wherein said causing a first one of said application servers step is further defined as

determining the server efficiency factors for at least two of said at least three application servers;

choosing the application server having the most desirable server efficiency factor to execute said first command set;

causing said chosen server to execute said first command set.

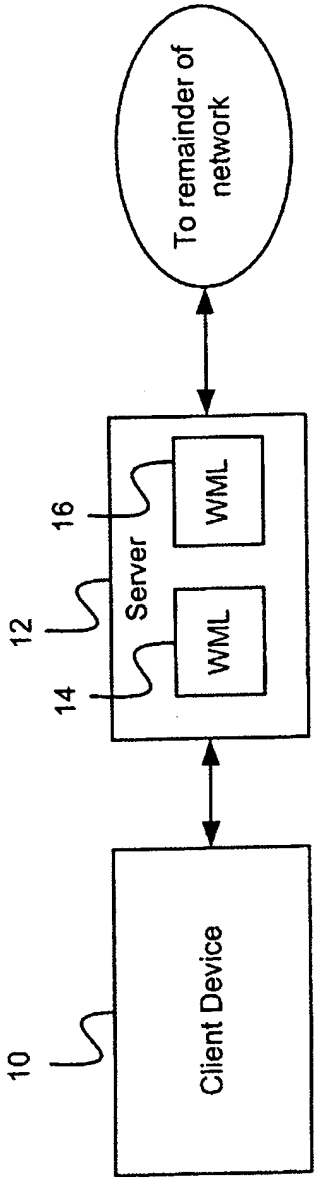


Fig. 1A
Prior Art

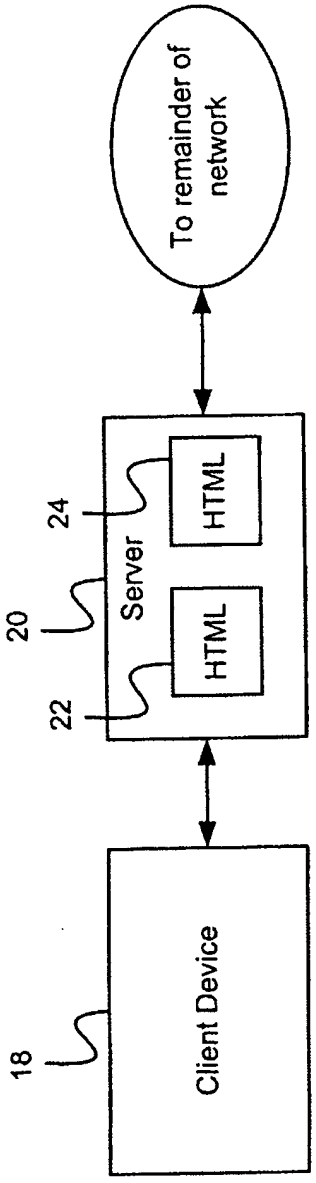


Fig. 1B
Prior Art

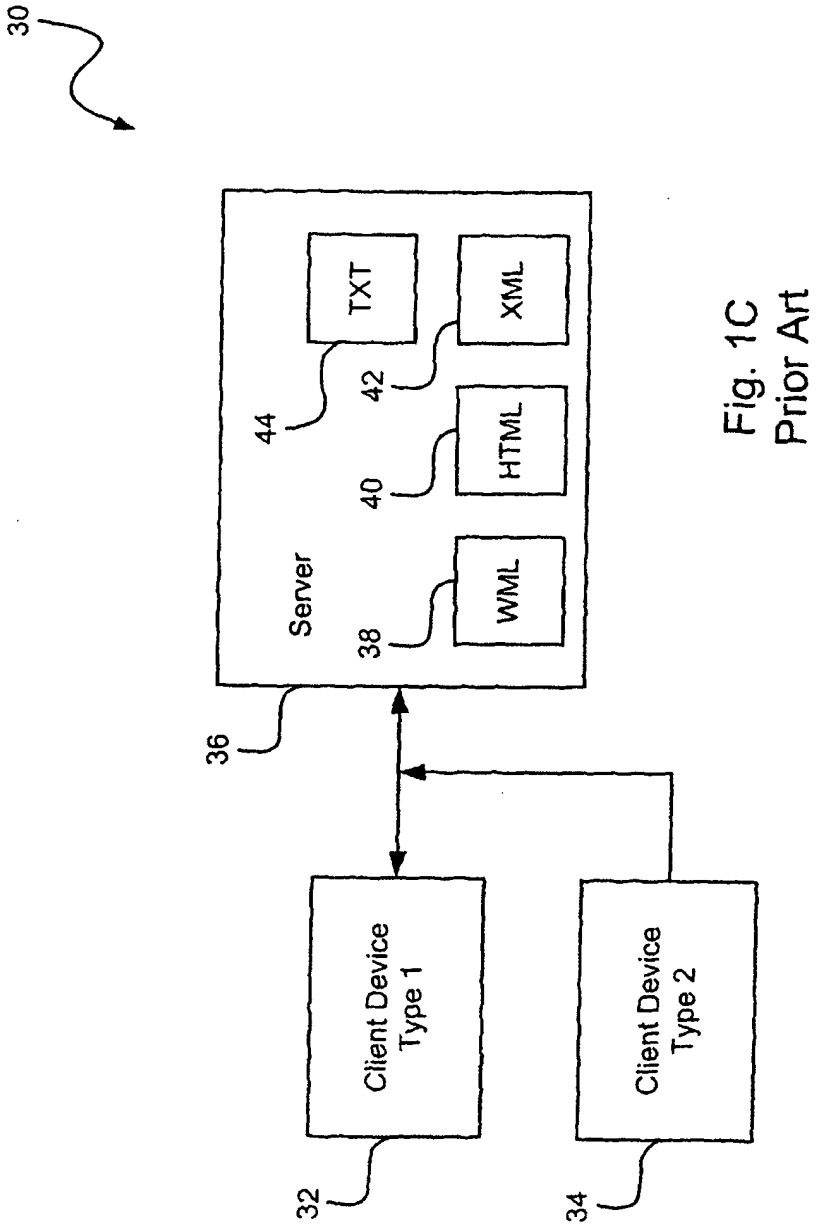


Fig. 1C
Prior Art

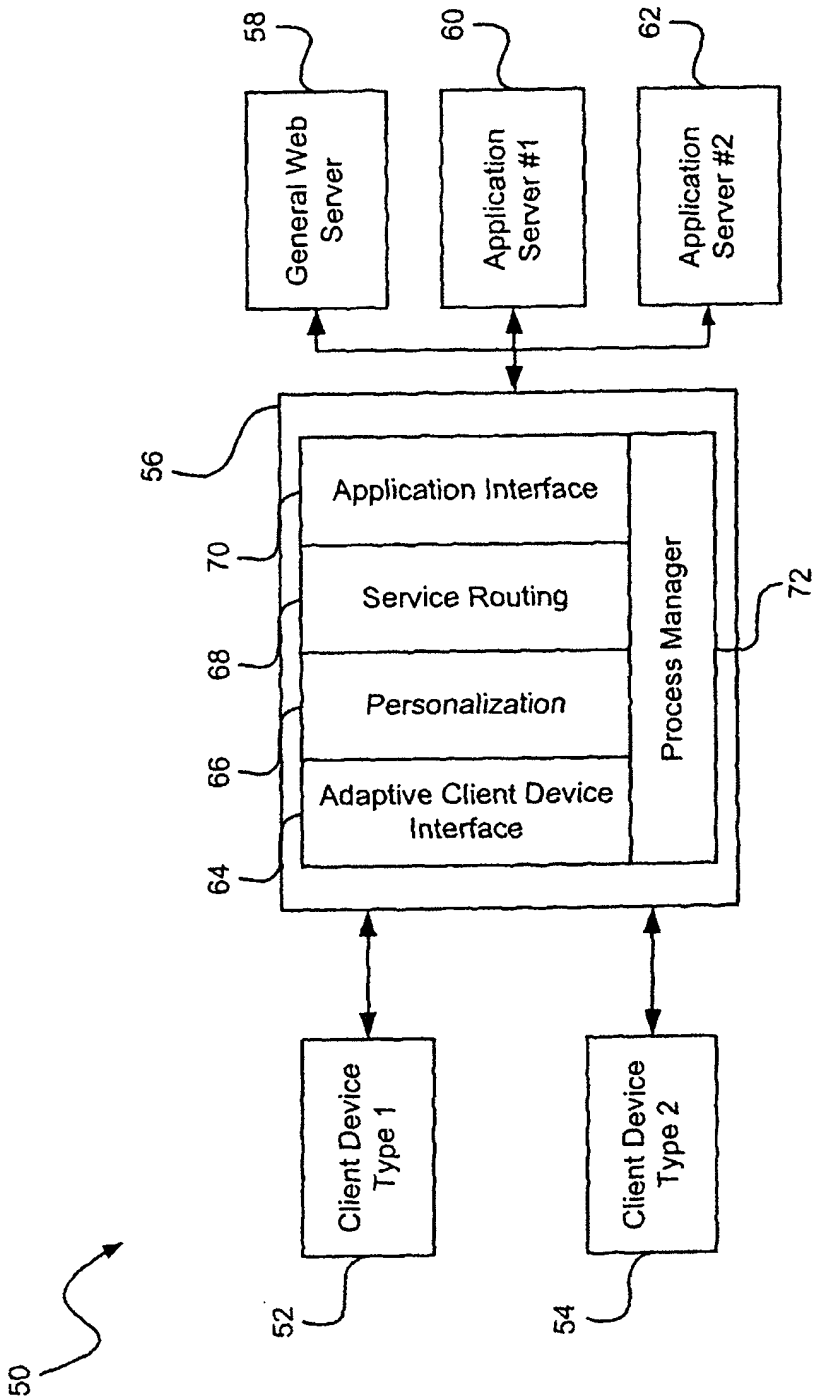

58
60
62
General Web Server
Application Server #1
Application Server #2
56
Application Interface
70
Service Routing
68
Personalization
66
Adaptive Client Device Interface
64
Process Manager
72
Client Device Type 1
Client Device Type 2
52
54
50

Fig. 2

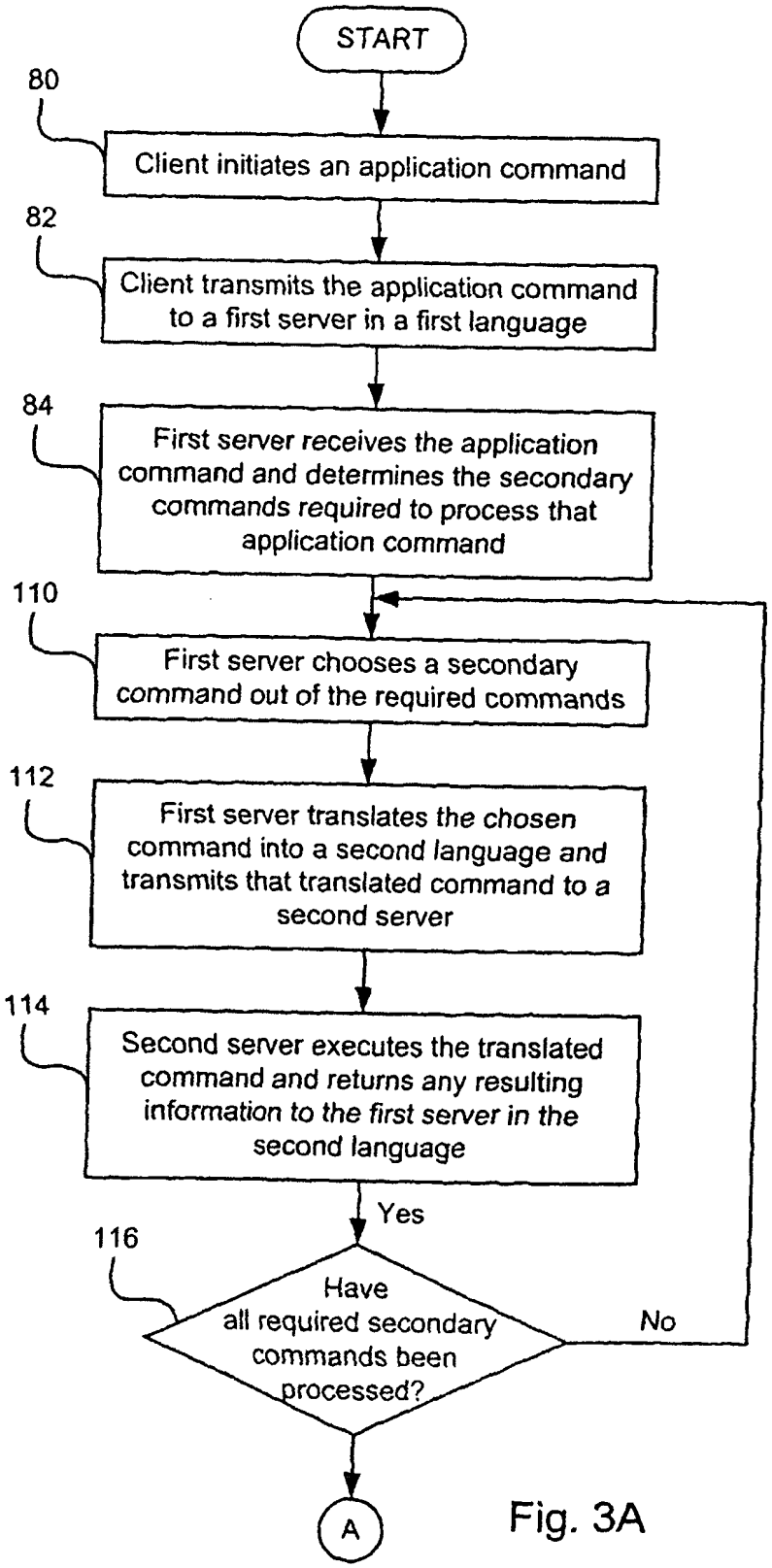

START
80
Client initiates an application command
82
Client transmits the application command to a first server in a first language
84
First server receives the application command and determines the secondary commands required to process that application command
110
First server chooses a secondary command out of the required commands
112
First server translates the chosen command into a second language and transmits that translated command to a second server
114
Second server executes the translated command and returns any resulting information to the first server in the second language
Yes
116
Have all required secondary commands been processed?
No
A

Fig. 3A

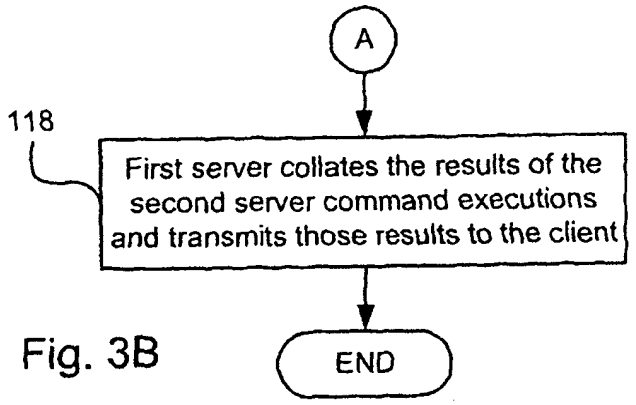

A
118
First server collates the results of the second server command executions and transmits those results to the client
END

Fig. 3B

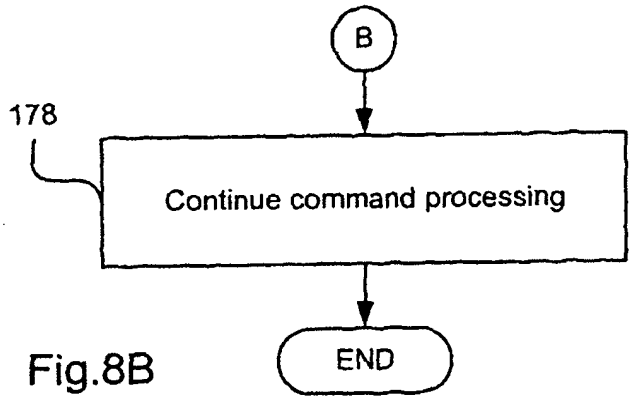

B
178
Continue command processing
END

Fig.8B

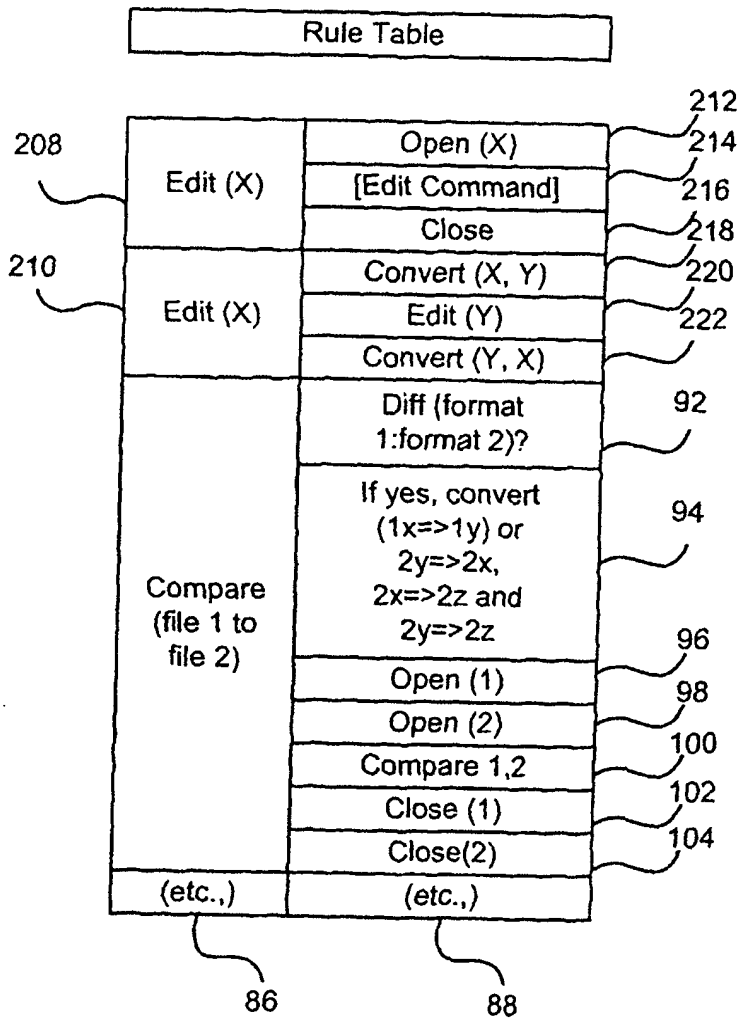


Fig. 4

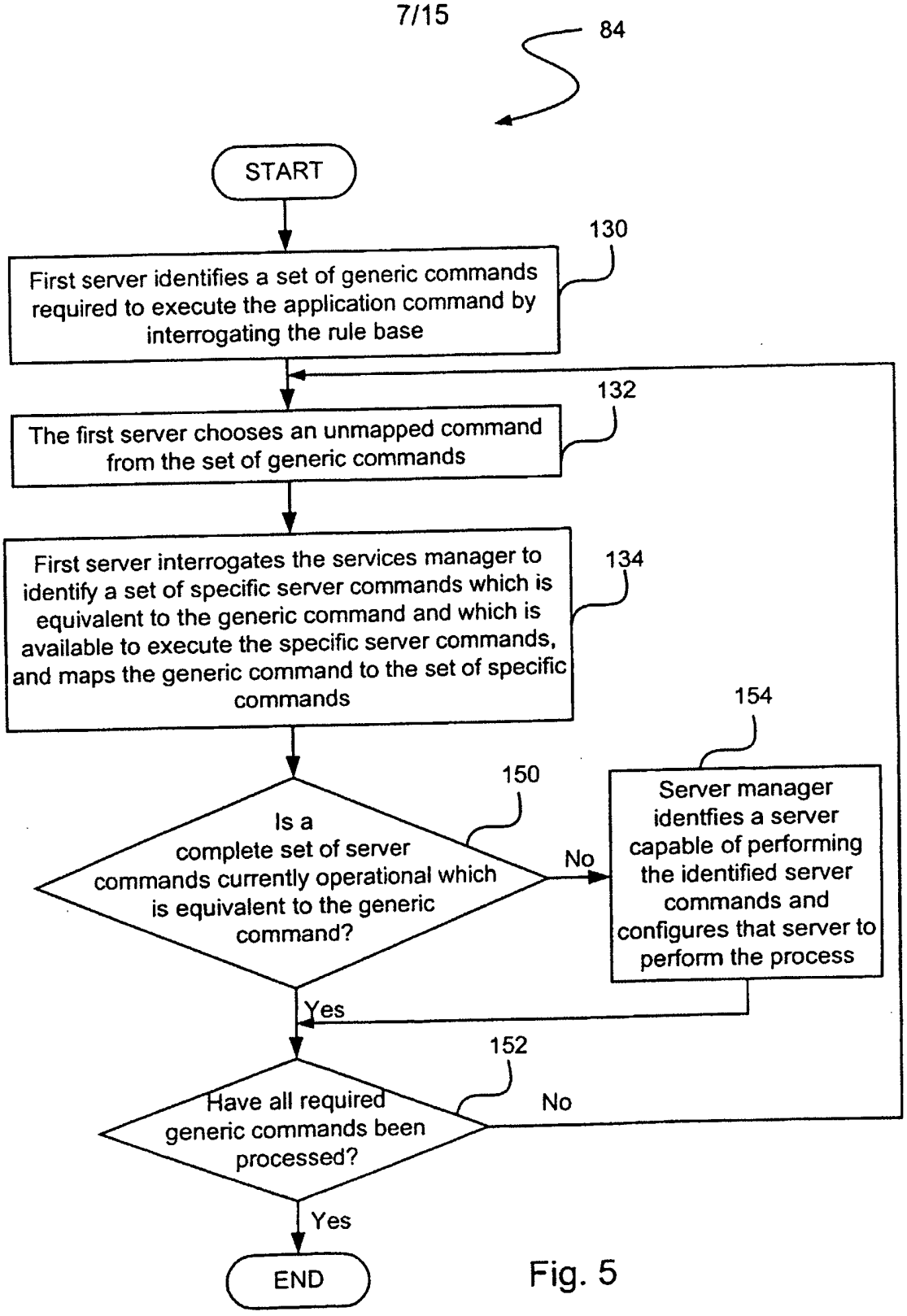

84
START
130
First server identifies a set of generic commands required to execute the application command by interrogating the rule base
132
The first server chooses an unmapped command from the set of generic commands
134
First server interrogates the services manager to identify a set of specific server commands which is equivalent to the generic command and which is available to execute the specific server commands, and maps the generic command to the set of specific commands
150
Is a complete set of server commands currently operational which is equivalent to the generic command?
No
154
Server manager identfies a server capable of performing the identified server commands and configures that server to perform the process
Yes
152
Have all required generic commands been processed?
No
Yes
END

Fig. 5

Capability Table		
Command	**Application**	**Server ID**
Add table	Word Processor #1	99
Delete Page	Word Processor #1	135
Add text	Word Processor #2	146
Compare (doc1, doc2)	Word Processor #3	135
Add Data	Spreadsheet Program #1	99
Search for information	Spreadsheet Program #2	99
Convert (x, y)	Convert	129
Convert (x, y)	Convert	135

136 138 140

Fig. 6

Server Loading Table	
Server ID	**Application Loading**
129	1.29
135	5.68
99	12.95
2405	13.74

Fig. 7

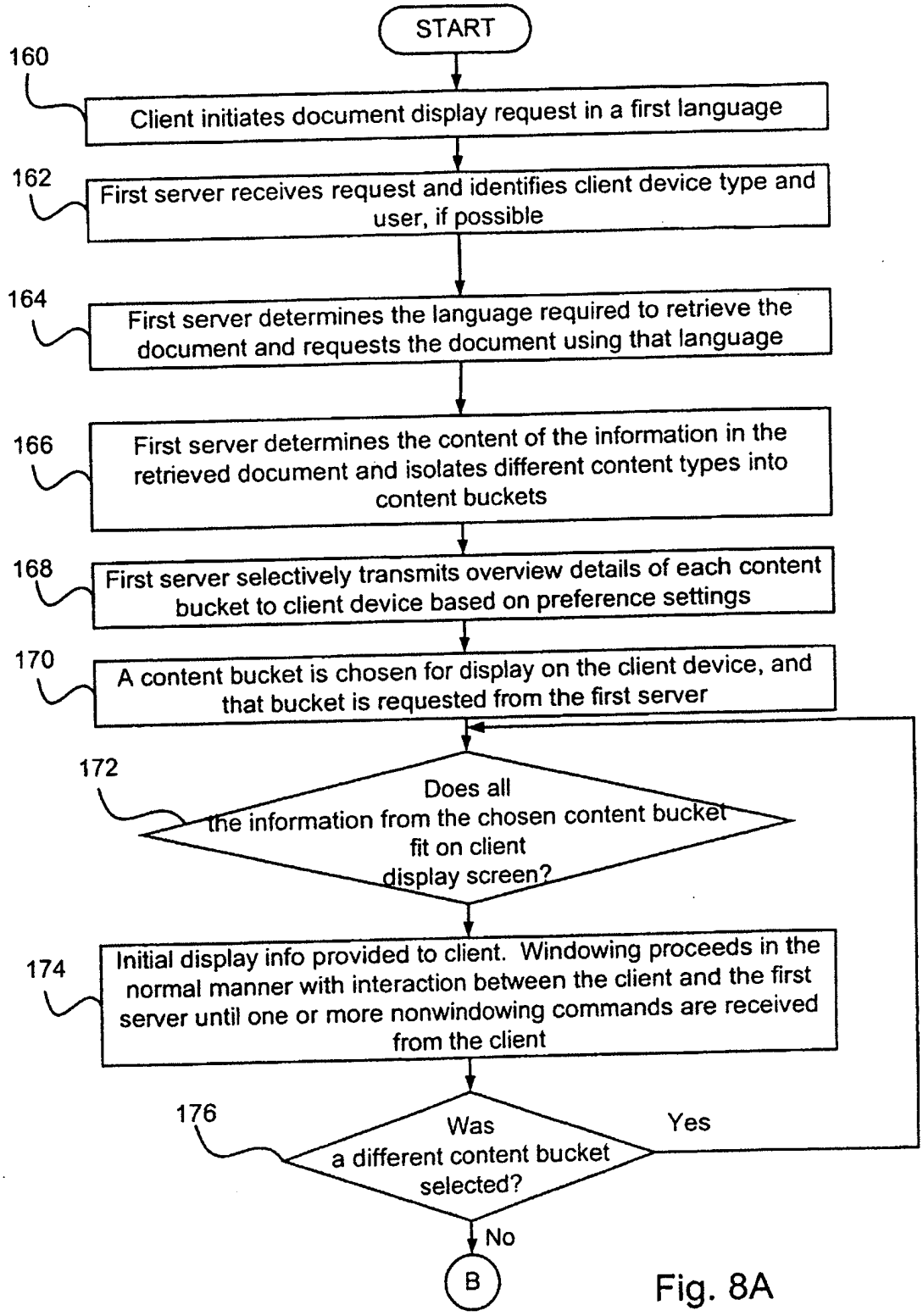

START
160
Client initiates document display request in a first language
162
First server receives request and identifies client device type and user, if possible
164
First server determines the language required to retrieve the document and requests the document using that language
166
First server determines the content of the information in the retrieved document and isolates different content types into content buckets
168
First server selectively transmits overview details of each content bucket to client device based on preference settings
170
A content bucket is chosen for display on the client device, and that bucket is requested from the first server
172
Does all the information from the chosen content bucket fit on client display screen?
174
Initial display info provided to client. Windowing proceeds in the normal manner with interaction between the client and the first server until one or more nonwindowing commands are received from the client
176
Was a different content bucket selected?
Yes
No
B

Fig. 8A

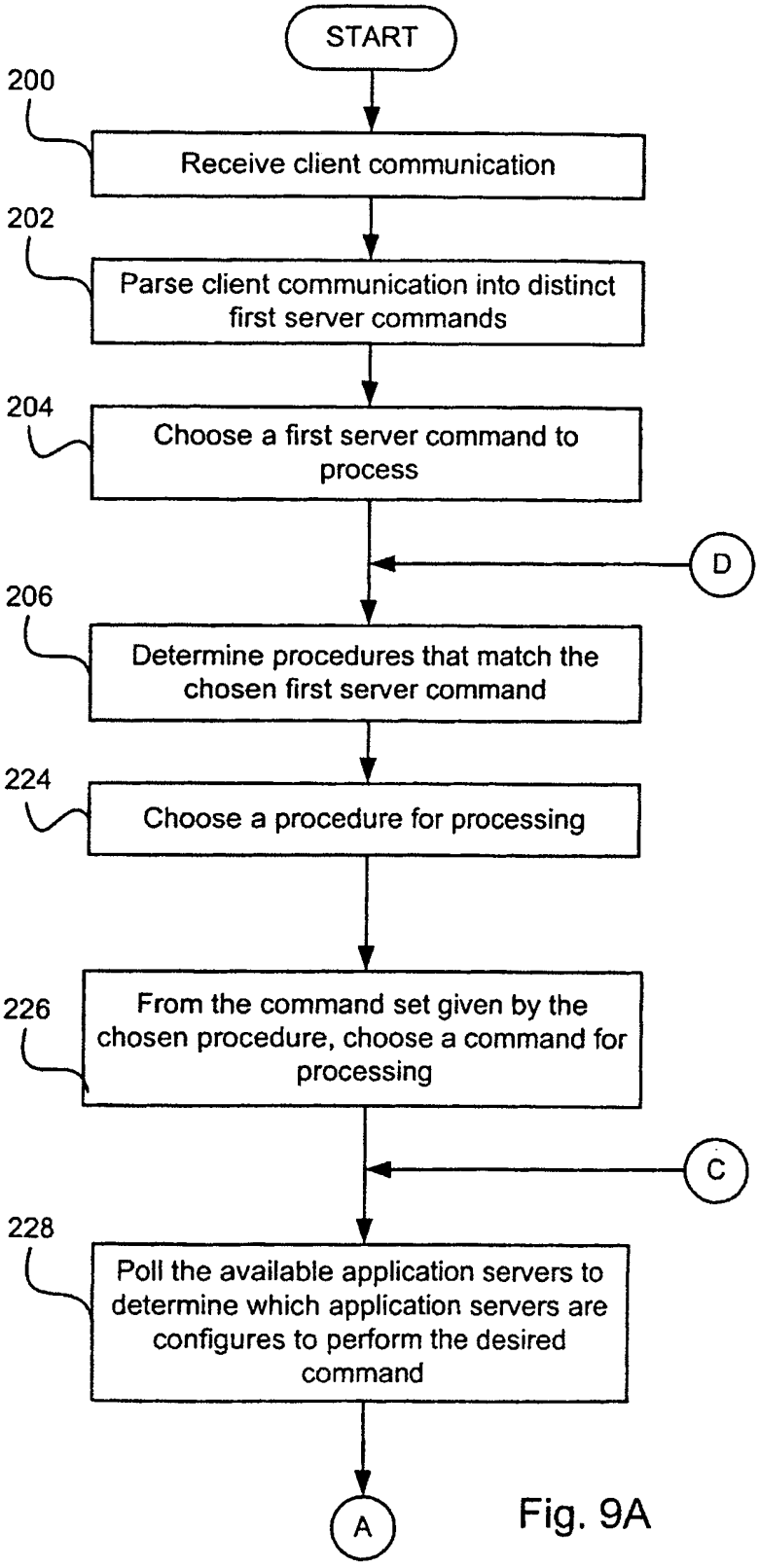


Fig. 9A

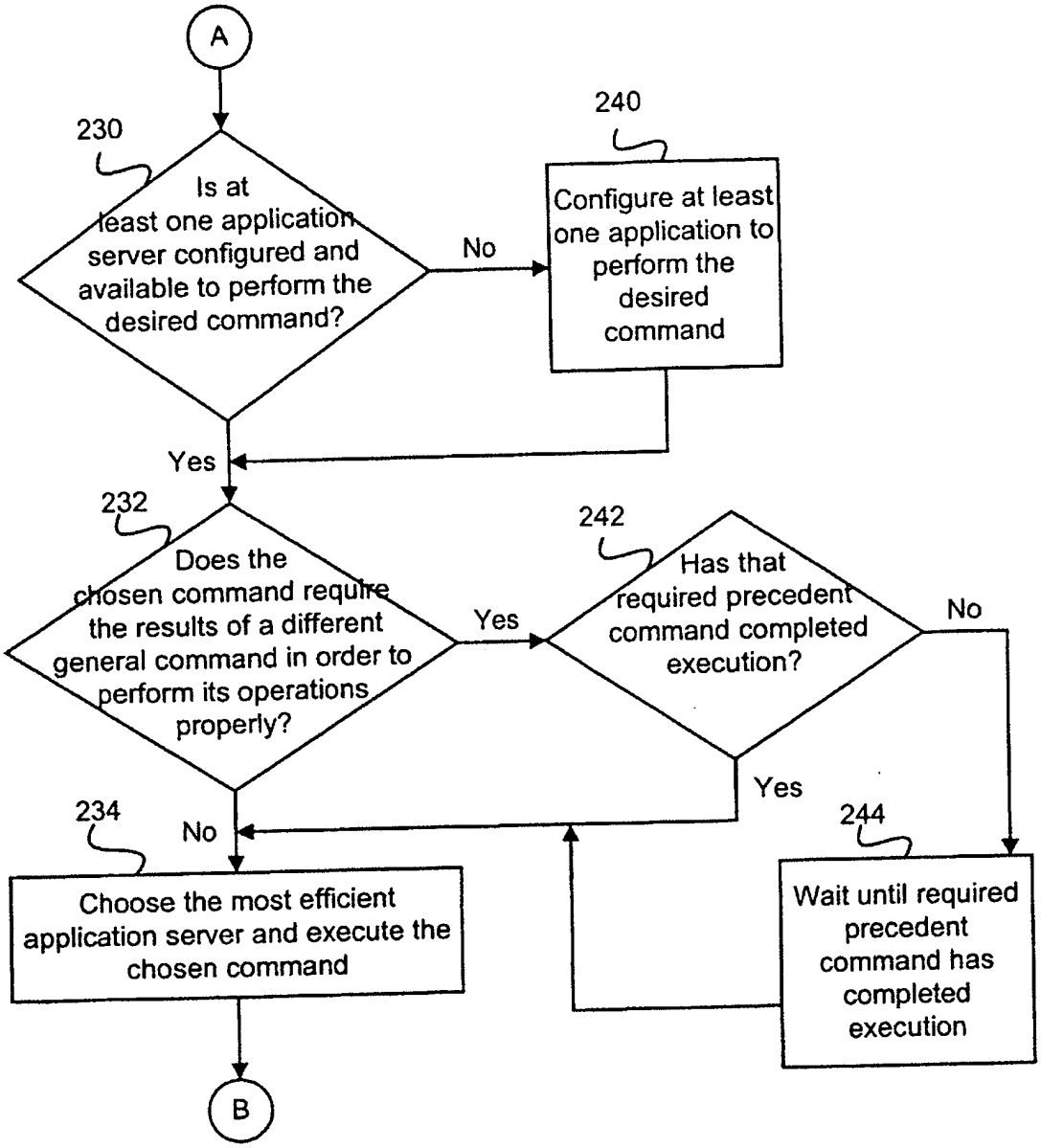

A
230
Is at least one application server configured and available to perform the desired command?
No
240
Configure at least one application to perform the desired command
Yes
232
Does the chosen command require the results of a different general command in order to perform its operations properly?
Yes
242
Has that required precedent command completed execution?
No
Yes
234
No
Choose the most efficient application server and execute the chosen command
244
Wait until required precedent command has completed execution
B

FIG. 9B

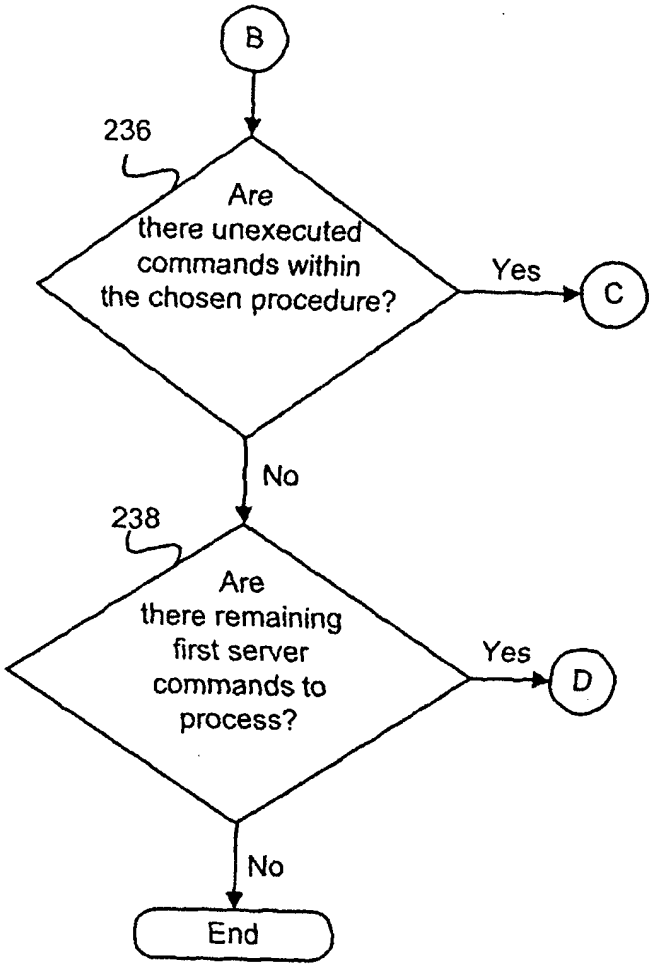


FIG. 9C

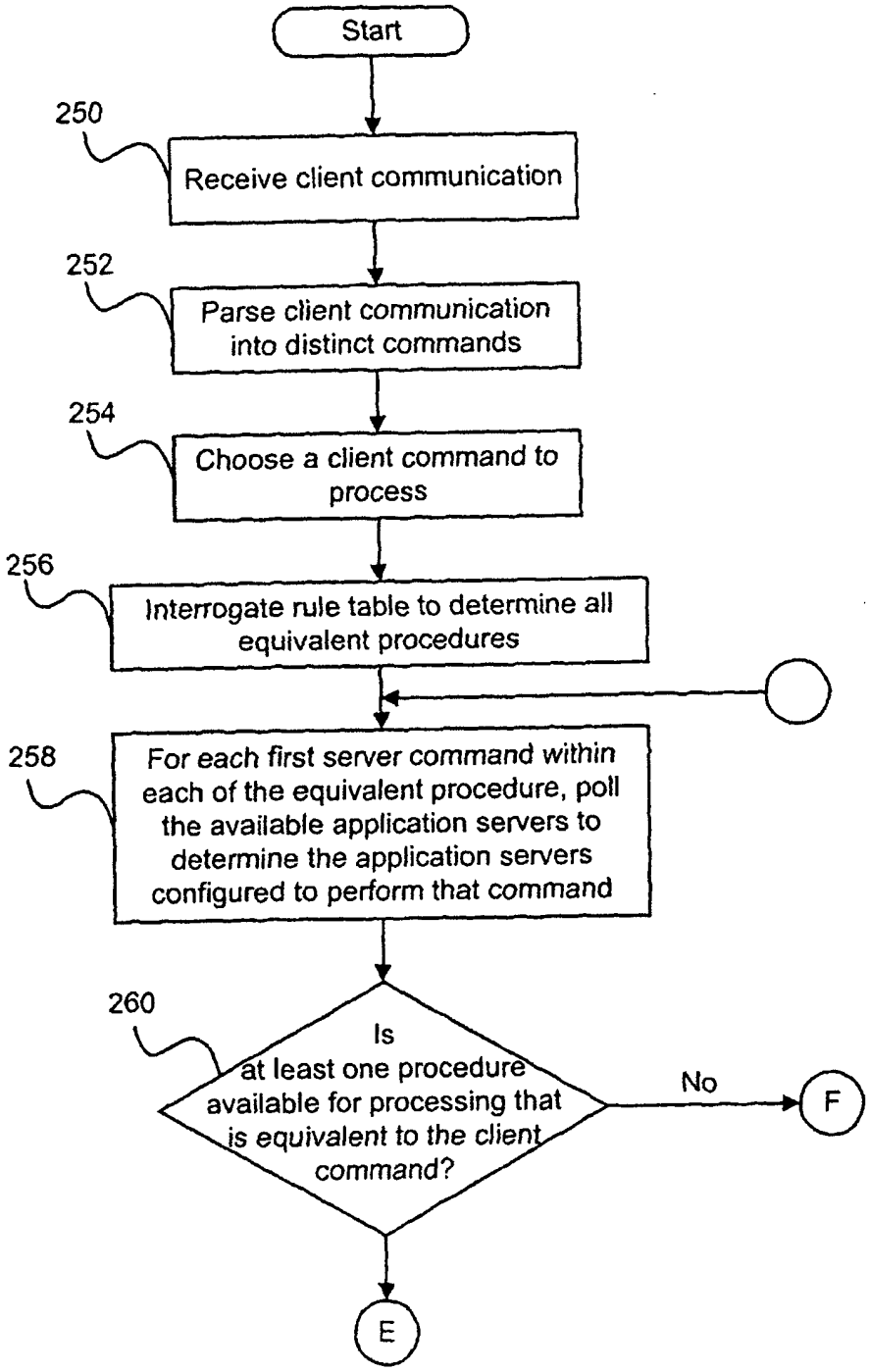

Start
250
Receive client communication
252
Parse client communication into distinct commands
254
Choose a client command to process
256
Interrogate rule table to determine all equivalent procedures
258
For each first server command within each of the equivalent procedure, poll the available application servers to determine the application servers configured to perform that command
260
Is at least one procedure available for processing that is equivalent to the client command?
No
F
E

FIG. 10A

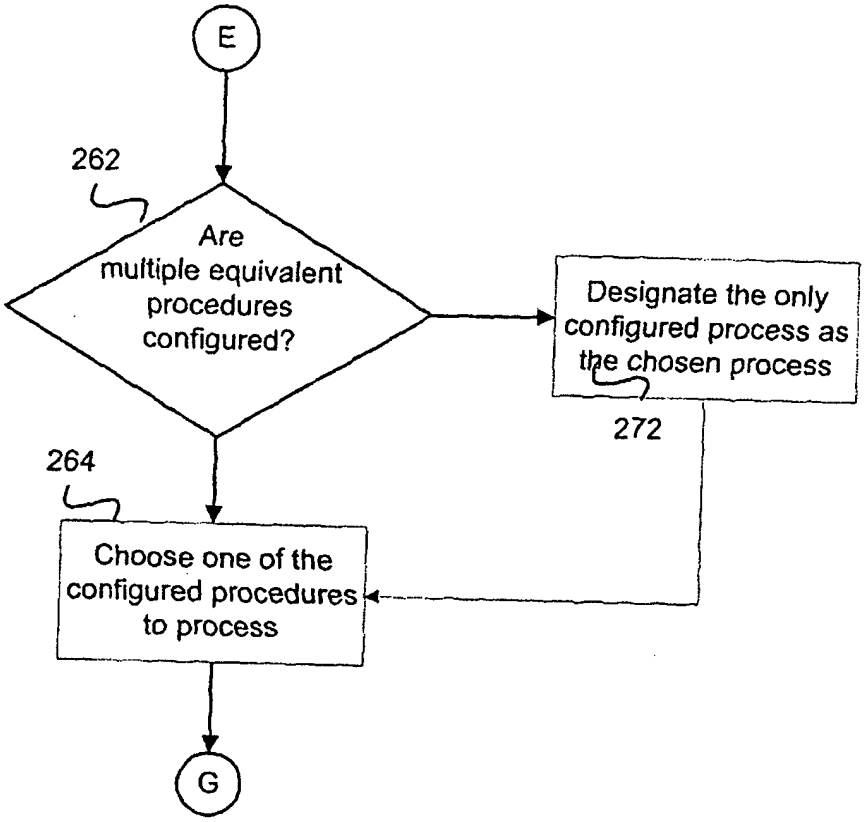


FIG. 10B

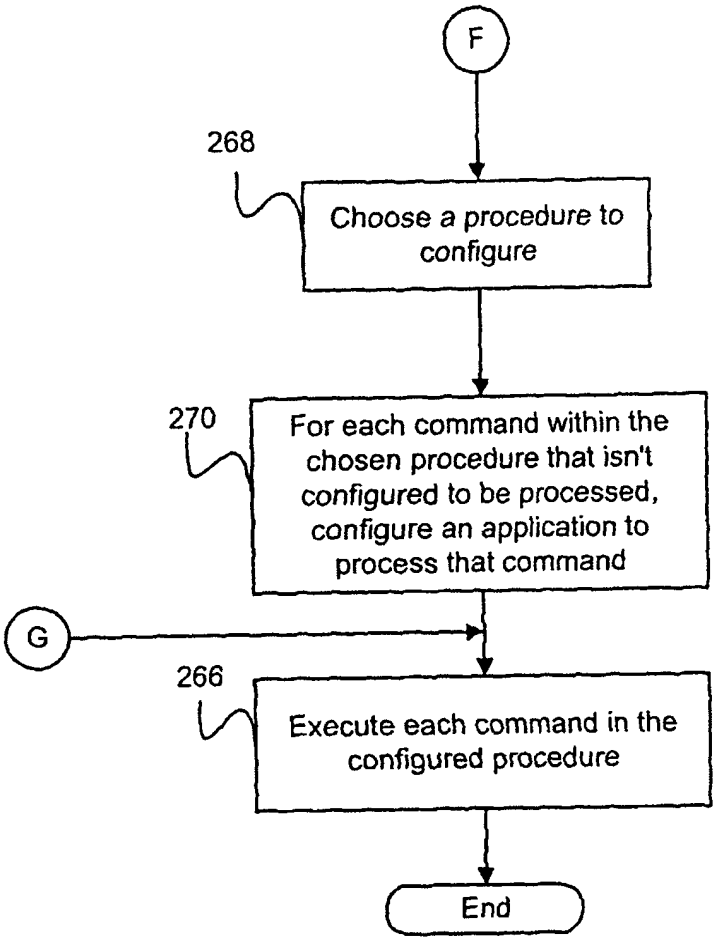

F
268
Choose a procedure to configure
270
For each command within the chosen procedure that isn't configured to be processed, configure an application to process that command
G
266
Execute each command in the configured procedure
End

FIG. 10C

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/06534

A. - CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 17/30
US CL :Please See Extra Sheet.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/9, 10; 717/5; 703/13, 20, 27, 28; 705/7; 709/201, 201, 217, 218, 228, 229, 316, 318

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

West/East, IEEE, CAS ONLINE, Dialog

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,590,200 A (EAGER et al.) 28 September 1999; see entire document.	1-15
Y	US 5,724,503 A (KLEINMAN et al.) 03 March 1998; see entire document.	1-15
Y	US 5,627,996 A (BAUER) 06 May 1997; see entire document.	1-15
Y	US 5,524,253 A (THONG et al.) 04 June 1996; see entire document.	1-15

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 APRIL 2001	10 MAY 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer JEAN M. CORR[illegible] James R. Matthews Telephone No. (703) 306-3035

Form PCT/ISA/210 (second sheet) (July 1998)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/06534

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

707/9, 10; 717/5; 703/13, 20, 27, 28; 705/7; 709/201, 201, 217, 218, 228, 229, 316, 318